TS Bulletin #3-92

Support
Your
Timex-Sinclair
User Group!

AUG. 18, 1992

Issue#3-92 --©Aug.,1992 in the public domain, photocopy freely --- L --- 8 pages

One does not expect a very fancy issue of any newsletter in the depths of the summer, but for this issue, we have saved a beautiful centrefold on the ZX-81/TS1000 internals especially designed for the programmer. It has been carefully checked, but there is a lot of information there to check, so if you notice any errors that have not been caught yet, please write us as soon as possible. This is sort of a pre-publication treat, for this illustration shall become a part of a book on the ZX/TS, to be distributed in photocopy form, earlier versions of which may already have reached your club lending library. Also for the programmer is the article on page 3 on hitchhikers, which offers a unique perspective on speeding up programs, both BASIC and m/c by tailoring any routine's function better to the structure of your computer's internals. Read also, on page 6, a controversial analysis of the structural changes in the economy including the job futures, in the second industrial revolution whipping through our industries now in North America, whether we like it or not.

A few words of explanation are perhaps useful for the article on page 2, the programmer's page. The BASIC listing there gives the complete routine for modifying a REM line anywhere in a BASIC program. It is easy to do it, if the REM is near the beginning of a program, but this routine allows you to save that valuable space at the beginning of your program for other things, such as loops that must run very quickly. It is set up to enter an LDOS command in a REM statement following the LDOS m/c disk access call, RAND USR 14336 at line 4545, by POKE of some ZX-81 CODE characters into that REM statement where otherwise there would be blank spaces (CODE=0). Of course the use of this need not be limited to manually entered text since you could POKE the text content of a string in there too by similar means. Other uses for such a routine as this would be to store some message inside the program itself, independent of any variables that could be cleared with CLEAR or RUN, on, for example the person or task the program was used for last (or even time it was run for). At the next retrieval of the program, a few PEEKs to this area could reveal the message stored in such a REM statement. Still other uses include password storage, and even making a program that can learn from previous experience. Anyway, enough tips are to be seen in this routine that it is worth printing for those not interested in its specific sample application of LDOS disk access from a running BASIC program. Too note that changing the characters in a REM statement is a simpler task than putting in a whole new line or one other than a REM statement, where differing line lengths would call for fancier programming still. (The writer is working on that one too).

Some problems are surfacing in the use of disks and other accessories with the older generations of personal computers including the ZX-81/TS1000 and some fixes and problem analyses are presented in the article on page 7. Page 8, talks about errors in print and corrections in our Timex-Sinclair computer hobby. Another fix, experimenting with many newsletters and hoping to move to book/booklet publishing is underway in publishing here. Nonetheless, there is no sign that the bottom of the in-basket of ZX/TS articles has been reached yet! Have a nice autumn. (If we get any hot weather it will have to get here soon; it has been extraordinarily cool!)

NEWS & NOTES:--It must be a slow summer when the postal system brings no new newsletters and even the junk mail volume is dwindling. Hello, out there in user group land. Is anybody there anymore? Only the Toronto TS Club has responded!
---A revision of the Unofficial LDOS Tech.Manual for ZX-81 is being undertaken here by the writer. A sample of some of the material is to be seen on p.2 and 7 here.

CONTENTS:
-Editing A REM Statement From Within A ZX-81/TS1000 BASIC Program . . .p.2
-ZX-81/TS1000 Internal Architecture For Programmers (Centrefold)... p.5-6
-Bug Reports And Other ZX-TS Ponderings p.8
-Disks, Cassettes And Modem Use With The TS And Other Classics p.7
-Programmers' Notes: Speeding Programs Using The Hitchhiker Idea p.3
-Jobs, Jobs, Jobs: Office Automation And The Birth And Death of Jobs .p.6
-Important Notices And Comments About This Newsletter, News, Etc. . . .p.1

This Just In:

Warning: Some errors may not have been caught in this. Please report any suspected promptly, especially as concerns the centre-fold which does deal with very complex and hard to check material and comes from a variety of sources.

PRODUCED AS A HOBBY ACTIVITY BY BILL J. HARMER, 97 RUSKIN, OTTAWA, ONT. CAN. K1Y4B3

Programmers' Corner: Changing A REM Statement From BASIC

This is an editor in BASIC that works much like the m/c LDOS editor for entering an LDOS command line. If you were to put in your program, GOSUB 4400 this routine would allow you to enter a BASIC line such as an LDOS command, as necessary to get it in as a disk command after that RAND USR 14336 call for LDOS. GOSUB 4545 would then make this call, although you would want your program also to do some checking as to the correctness of the command before turning it over to LDOS, since any error results in the 'real' LDOS screen being activated. This shows you roughly how the LDOS screen edit for LDOS command line works as well. Lines 4520-4530 use a trick useful to learn for other uses.

```
4399 REM FOR ZX-81/TS1000 Only
4400 REM THIS INITILIZATION IS NEEDED IN YOUR PROGRAM SOMEWHERE (LINES 4400-4490)
4410 SLOW
4490 LET Y=0
4500 REM START OF ACTUAL EDIT ROUTINE TO MODIFY REM LINE 4550
4505 IF Y>19 OR Y<0 THEN LET Y=0
4510 PRINT AT Y,0;"                   ";AT Y+1,0;"*";          4511 REM PRINT * PROMPT
4515 LET X$=""         (4517 LET A$="                   "  4518 REM 19SPC,SEE4650)
4520 LET N=5                                    4525 REM lines next must not change
4530 LET NA=28+PEEK 16425+256*PEEK 16426
4540 GOTO VAL"4560"
4545 RAND USR VAL"14336"
4550 REM                                         (note that a lot of spaces here)
4555 REM CHANGE N IN LINE 4520 TO 10 AND ALL OTHER OCCURRENCES OF 5 TO 10 IF
     THIS ROUTINE TO START WITH FIRST 5 CHARS. IN REM AT 4550 SET TO FOR EXAMPLE
     PERMANENTLY TO SOMETHING, LIKE LOAD" AND ALSO MAY WANT TO CHANGE 19 IN LINE
     4620 TO 18 IF YOU DO THIS-NA=address for POKE, of first char.after REM(4550)
4557 RETURN                        4558 REM RETURN FOR RAND USR 14336 CALL see top
4560 LET X=0                       4561 REM NOTHING ELSE MUST GO FROM 4530 to 4550---
4570 LET X$=INKEY$                 4571 REM NOTHING- NOT EVEN ONE BYTE MORE OR LESS
4580 IF X$="" THEN GOTO 4570
4590 LET X=CODE X$
4595 IF X=119 AND N>5 THEN PRINT AT Y+1,(N+1-1-5);CHR$(128+PEEK (NA+N))
4596 REM THAT LINE WILL INVERT ON BACKSPACE LIKE LDOS--WHICH IS A BIT CRUDE,
         BACKSPACE=RUBOUT OR DELETE ON ZX-81/TS1000 KEYBOARD, remove -1 AFTER
         N IF YOU DO NOT USE A PROMPT LIKE THE * USED HERE AT BEGINNING CMD LINE
         FOR EXAMPLE, ALTHO OTHER CHANGES NEEDED IN CODE TO DO THAT TOO
4600 IF X=119 AND N>5 THEN LET N=N-1          4601 REM BACKSPACE POSITION IN LINE
4610 IF X=119 THEN GOTO 4560    4611 REM MAKES NOT ENTRY OR CHANGE IN INPUT
4620 IF N>19 OR X>63 OR Y>20 THEN GOTO 4698  4621 REM IF X > 63 catches ENTER-N/L
4622 REM IF X>63 IS ALSO A CRUDE ERROR REJECTOR, Y>20  IS NOT GOOD FOR BOT.SCRN.
4630 PRINT AT Y+1,(N+1-5);X$           4631 REM REMOVE THE 1 IF REV., See 4555,PRMPT
4635 REM THIS IS THE PRINT THAT PUTS THE CHARACTER KEYED-IN ON SCRN.,(ECHOS IT)
4640 POKE NA+N,X
4645 REM THIS POKES THE CODE INTO THE REM STATEMENT ON LINE 4550
(4650 LET A$(N TO N)=X$      4655 REM THIS MIGHT BE USED TO PUT THE COMMAND INTO
        A STRING A$ SO THAT IT COULD BE ANALYSED/PARSED-ERROR CHECKED ELSEWHERE,
        SEE ALSO LINE 4517 ABOVE)
4660 LET N=N+1                    4661 REM MOVES POSITION POINTER, CMD.STRING/LINE FWD.
4670 GOTO 4560
4698 LET Y=Y+2         4697 REM INCREMENTS PRINT LINE LIKE LDOS SCROLLING
4699 RETURN     4700 REM THIS ENDS EDIT ROUTINE CALLED AS SUBROUTINE TO EDIT 1 LINE
```

……………………………

This has been typed from a tested program with some elaboration, mainly extra REM statements to clarify its operation principles. Use of a routine like this will help in putting LDOS commands into a running program and as a bonus listing it gives the reader a hint as to the steps that the LDOS line editor for LDOS CLINE (command line) goes through. The LDOS CLINE editor is in m/c though.

A routine like this could be used to build a utility program for use with LDOS or a BASIC or part-BASIC operating system shell, etc. A few lines added to translate from a completely different command language into LDOS command language, poked into the REM statement is another approach to this, so that COPY could be translated into two LDOS commands, LOAD"---name of file, and SAVE"---name of file, emulating CP/M or MS DOS style of command language!

Many other uses include putting a secret password or time out number in REM where it is safe from being cleared by CLEAR or RUN for any system.

Hitchhikers Speed The Job - Use Some Hitchiking Bits As A Flagman

You can speed up the limited processing power of your computer by a method of using a register for more than one 'thing' at a time. We may call this extra use of it, for both the main data item and an extra one, a hitchhiker, so to speak. A little example: To compress a text (of characters), the most frequently occurring character in any English or other natural language is the space. It would be a natural to represent in a smaller code than 8-bits (as in ASCII, etc. alphanumeric character codes). Since in text, you usually don't need the codes 128-255, you can use 7-bit codes for all the letters. To indicate that a space precedes a letter, you simply set the MSbit of the byte to 1 (reset it to zero if not). Since the average microprocessor uses 8-bit registers, and so does BASIC character string manipulation, this MSbit can be thought of as a hitchhiker, that rides along with the alphanumeric code, when set indicating a space precedes the letter the code stands for (in ASCII for example). When you are searching text for a word, nearly always it will begin with a space. Searching a compressed text for a byte which has the right 7-bit code and the MSbit set, to indicate a space preceding, means that you can speed up the search process, which would normally take two steps to get that far, checking for a space, and then branching to a routine to check to see if the next letter is the one you are looking for a word beginning with. Since a space normally occurs each six characters in English on average, your routine, BASIC or machine code, will save going into this second loop to check the second letter, an average of 1/6 the time or perhaps saving the time of going into this loop unnecessarily about 13-16% of the time, reducing the time searching for the word where it isn't there, 2 or 3 times that (33% plus). A routine that speeds up word search about 33% and saves about 16% of memory through compression of the text data in addition looks like a win-win situation indeed. And it is all based on having that extra MSbit flagging a preceding space or representing a second letter when it is a space, to put it more exactly, hitchhiking on the 8-bit register operations in machine code and 8-bit character code handling of BASIC if you write the routine in BASIC even. And control codes can use it.

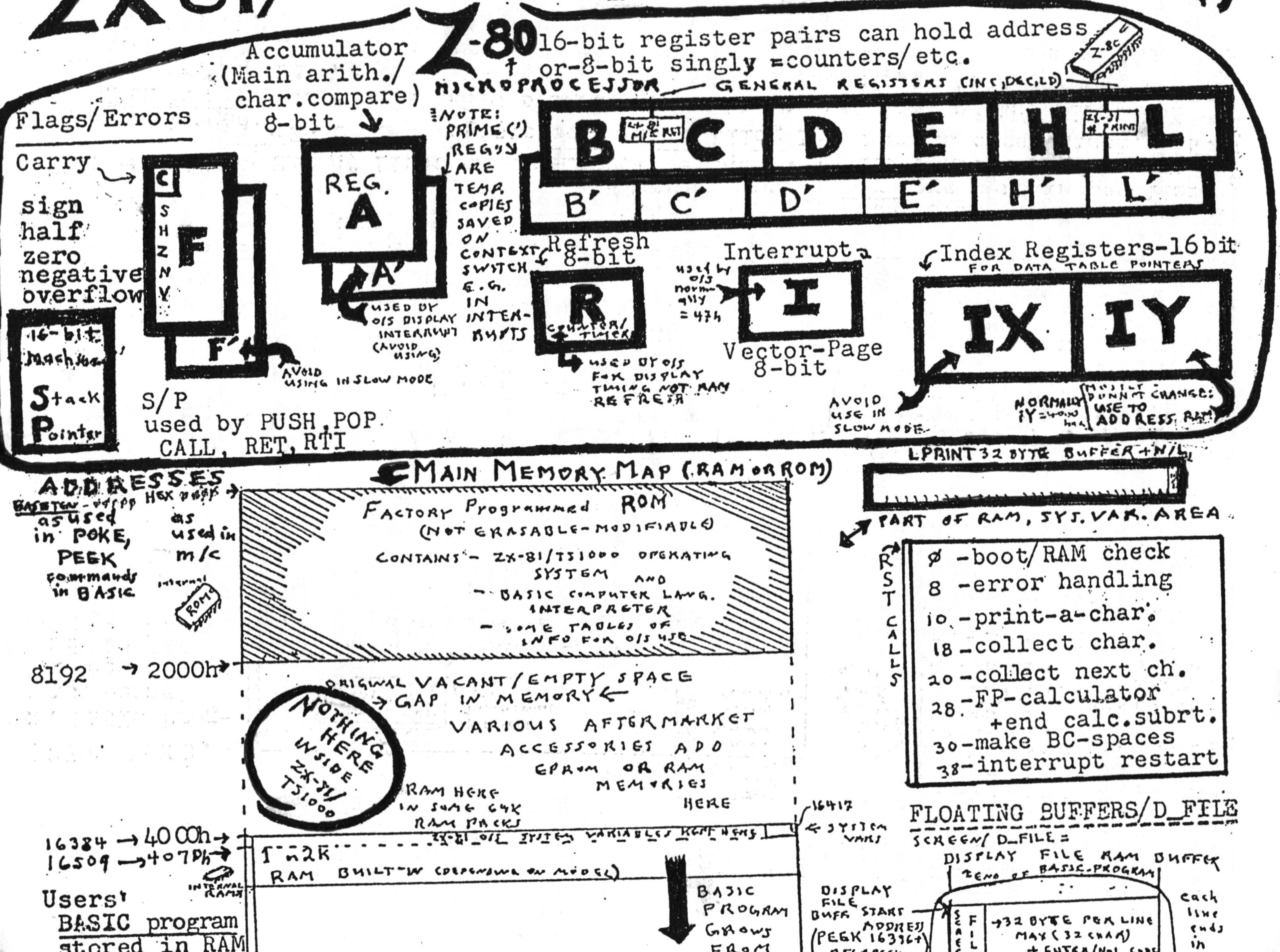
Of course this hitchiking is not a new idea. One of the first of the regiter computers, the IBM 360 with its 32-bit registers gave some programmers the idea in writing a compiler. If you compile a math routine into a series of m/c calls to do adding, and then put the numbers in RAM, then get one out again and do some subtraction, but it in RAM, then get it out again and add one if flag so-and-so is set, then put the result in RAM take it out ... it all adds up to wasting a lot of time putting the number in RAM and taking it out again to put in a fast register. You could use a two pass compiling procedure that puts dummy commands in and only decides on the second pass, after the compiler examines the next step, whether the arithmetic operation it is to convert into machine code should end by putting the number into RAM, or if it will be used again, keeping it in the same register. But compiling is awfully slow anyway, so the programmers decided that there was a better way. Only part of the 32-bit register needed to be used for the object being worked on, but some more of the left over bits could be used to hold a code or system of flags, telling the compiler, on the next step, whether the result left in that register should be put in RAM or left there, and generate the right code to do it. The result would be optimized arithmetic in the compiled program and using other parts of one register would result in little slowing of the compilation routine itself. (For the real, detailed story, p.364-6, Gries, "Compiler Construction for Digital Computers", Publ.1971, John Wiley.)

So you can use the extra space you can find or make in a register to make a routine run faster, the extra flag or data items, or code, hitchhiking along in even an 8-bit register. For 16-bit or 32-bit registers, even more is available, so that complicted codes can be used. Clever use of certain flags like carry and zero might also be used, as in the search through a text example, encountering a space before the letter might be flagged by setting a flag not normally set when a space is detected, and clearing it if the next character is not the letter searched for as the second character (first letter after the space) of the word you are searching for. A lot of register operations using 8-bit registers (or 16) can be treated as having an extra bit or two if you make clever use of the flags of the microprocessor as if they were extra 1-bit registers, when you are machine code programming. Also, BASICs often use 16-bit integers (no byte arrays available) so when unpacking text into numbers, that should say to you, can the extra bits be to be used as a hitchhiker?

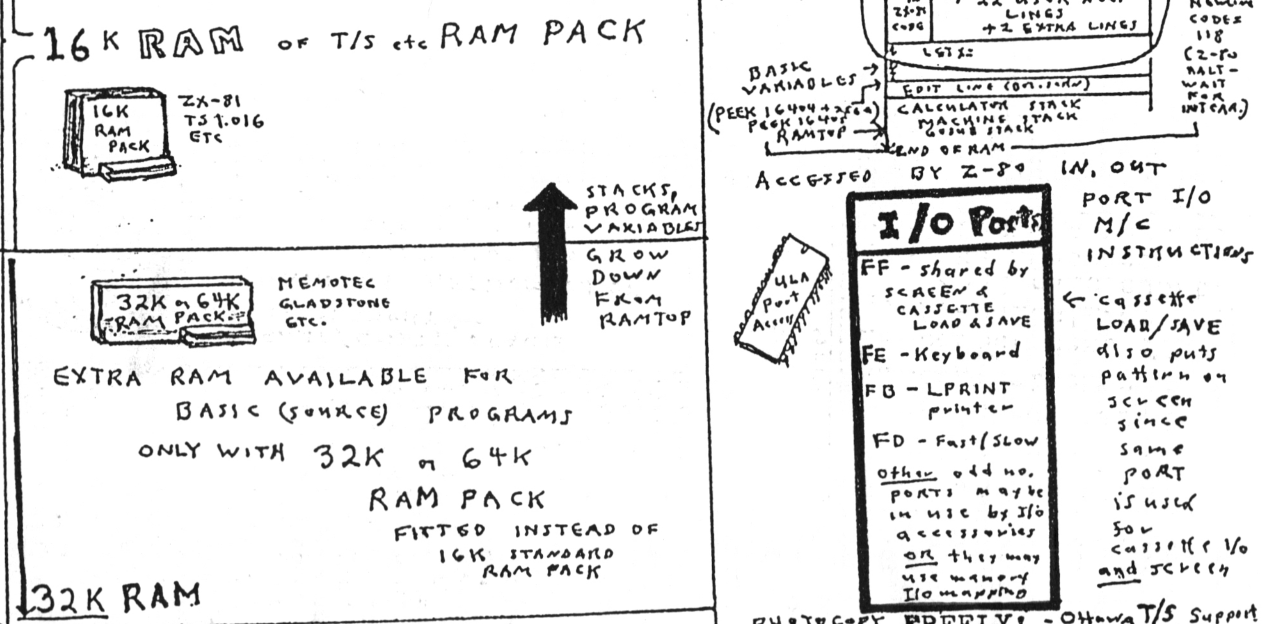
VER. 1.0.91

PROGRAMMERS' MAP OF THE

ZX-81/TS1000 HOME COMPUTER


Z-80
MICROPROCESSOR
16-bit register pairs can hold address or-8-bit singly =counters/etc.
Z-80
GENERAL REGISTERS (INC,DEC,LD)
B
C
D
E
H
L
B'
C'
D'
E'
H'
L'
ZX-81 M/C RET
ZX-81 # PRINT
Accumulator (Main arith./ char.compare) 8-bit
REG. A
A'
USED BY O/S DISPLAY INTERRUPT (AVOID USING)
NOTE: PRIME(') REGS ARE TEMP. COPIES SAVED ON CONTEXT SWITCH E.G. IN INTER-RUPTS
Flags/Errors
Carry
sign
half
zero
negative
overflow
C
S
H
Z
N
V
F
F'
AVOID USING IN SLOW MODE
16-bit machine
Stack Pointer
S/P used by PUSH, POP CALL, RET, RTI
Refresh 8-bit
R
COUNTER/TIMER
USED BY O/S FOR DISPLAY TIMING NOT RAM REFRESH
Interrupt
I
used by O/S normally = 47h
Vector-Page 8-bit
Index Registers-16bit
FOR DATA TABLE POINTERS
IX
IY
AVOID USE IN SLOW MODE
NORMALLY IY=4000h
MOSTLY DON'T CHANGE: USE TO ADDRESS RAM
LPRINT 32 BYTE BUFFER + N/L
PART OF RAM, SYS. VAR. AREA
RST CALLS
Ø -boot/RAM check
8 -error handling
10 -print-a-char.
18 -collect char.
20 -collect next ch.
28 -FP-calculator +end calc.subrt.
30 -make BC-spaces
38 -interrupt restart
MAIN MEMORY MAP (RAM OR ROM)
ADDRESSES
BASE TEN - 0000 HEX 0000
as used in POKE, PEEK commands in BASIC
as used in m/c
internal ROM
FACTORY Programmed ROM (NOT ERASABLE-MODIFIABLE)
CONTAINS - ZX-81/TS1000 OPERATING SYSTEM AND
- BASIC COMPUTER LANG. INTERPRETER
- SOME TABLES OF INFO FOR O/S USE
8192
2000h
ORIGINAL VACANT/EMPTY SPACE
GAP IN MEMORY
NOTHING HERE INSIDE ZX-81/TS1000
VARIOUS AFTERMARKET ACCESSORIES ADD EPROM OR RAM MEMORIES HERE
RAM HERE IN SOME 64K RAM PACKS
16384 → 4000h
16509 → 407Dh
16417
SYSTEM VARS
ZX-81 O/S SYSTEM VARIABLES KEPT HERE
1 or 2K RAM BUILT-IN (DEPENDING ON MODEL)
INTERNAL RAMS
Users' BASIC program stored in RAM
BASIC PROGRAM GROWS FROM
FLOATING BUFFERS/D_FILE
SCREEN/D_FILE = DISPLAY FILE RAM BUFFER
END OF BASIC-PROGRAM
DISPLAY FILE BUFF START ADDRESS (PEEK 16396+
32 BYTE PER LINE MAX (32 CHAR)
+ ENTER/N/L CODE
each line ends in

tokenized form)


16K RAM of T/S etc RAM PACK
16K RAM PACK
ZX-81 TS1016 ETC
STACKS, PROGRAM VARIABLES
GROW DOWN FROM RAMTOP
32K or 64K RAM PACK
MEMOTEC GLADSTONE ETC.
EXTRA RAM AVAILABLE FOR BASIC (SOURCE) PROGRAMS
ONLY WITH 32K or 64K RAM PACK
FITTED INSTEAD OF 16K STANDARD RAM PACK
32K RAM
ZX-81 CODE
LINES +2 EXTRA LINES
NEWLINE CODES 118 (Z-80 HALT-WAIT FOR INTERR.)
BASIC VARIABLES
EDIT LINE
CALCULATOR STACK
MACHINE STACK
GOSUB STACK
(PEEK 16404+256*
PEEK 16405)
RAMTOP
END OF RAM
ACCESSED BY Z-80 IN, OUT
PORT I/O M/C INSTRUCTIONS
ULA Port Access
I/O Ports
FF - shared by SCREEN & CASSETTE LOAD & SAVE
FE - Keyboard
FB - LPRINT printer
FD - Fast/Slow
other odd no. ports maybe in use by I/O accessories OR they may use memory I/O mapping
← cassette LOAD/SAVE also puts pattern on screen since same PORT is used for cassette I/O and screen

32768 →8000h

*A BASIC Program being expanded from 16K to 32K (when larger RAM pack in use) must not let D_File straddle 32K border

49152 → C000h

Any attempt to expand a BASIC program beyond 32K in size (with 64K RAM) will make computer crash!

-With a 64K RAM Pack extra memory is available here, but only for special uses, data saving, not for programs any bigger than 32K in any case.

-May include useless mirror image areas of RAM here.

Use of any further pages rare indeed

65535 → FFFFhex End of first page memory

†Z-80 is a registered name of Zilog Inc.

PHOTOCOPY FREELY! - Ottawa T/S Support
© Aug. 1992 in the public Domain

Further Reading: Sync Mar. 84, & Syntax (SQ), Winter 1982 magazines in your user gp. library perhaps ....also book "Explorers Guide to ZX-81" by Mike Lord, (British)

Note: Some non-TS or Sinclair my have certain peculiarities affecting use in certain cases, incl. 16K.

* Care Needed:
When you are writing a BASIC program, when it is just filling the first 16K and getting to the point where it is starting to be bigger than a 16K RAM pack equipped computer can hold, you must avoid the gradual expansion that puts the D_File accross this boundary. (Duplicating a giant REM is usual fix.)

Office Automation: The Second Industrial Revolution?

Yes we did survive the industrial revolution and our democracy became even more democratic at the same time. But will the middle class survive this second revolution of technology, the automation of information used to control economic activities at every level of production, factory robots, retail inventory control and sales and the automation of most office jobs as well? Let's start with the examination of the middle class. It is literally the class between the rich whose comfort and security is of a high level and the lower class, largely employed if at all only intermittently to which we add the working class, employed in one, steady job, but on an hourly basis. Many jobs that used to be considered working class because of the hourly fluctuations of emplyment and low wage levels became lumped in with the middle class as fluctuations in working became less and wages higher in the prosperity of this century. Many other jobs as well as part of the working class sort of manufacturing jobs, also became part of the middle class as the need to function on a level of literacy and number handling, mainly accounting, in so many jobs, started to merge with that necessary in middle class business/office jobs. In this century the pay for such jobs and the number of such jobs vastly increased, making the prosperity of the twentieth century seen to be based on this expansion of our middle class. Then came the computer revolution, in which the craft-like, customized handling of numbers and words that this middle class prosperity was based on is being done by mass production and by another machine, the computer, linked in networks, so that not even an office boy, much less a whole level of instruction and order carrying and carrying out managers, need trot between nodes in the network, factory/office with a bundle of papers under their arm. The resultant shrinking of the middle class has led to a spiral downward of the pay of those remaining in it, as larger numbers compete for fewer jobs. An article in Scientific American magazine, of Feb. 1992, p.111, documents the exodus of numbers from the middle class, mostly down, not up.

Personally, this writer disbelieves that we are seeing the effect of a post-industrial society* where fewer are needed to manufacture goods, and more needed to create or process information. That process has only slowly started to shift into effect and instead, the contracting out of labour-demanding manufacturing to low wage countries has preceded it. The fact that jobs may be lost due to increases in efficiency and pay money lost then also being temporary, may have been true in a time that loss of manual manufacturing jobs was replaced by craft-like and manual work of information processing, but in this cycle, it is the middle-class information handling jobs that are disappearing at the same time. They may be replaced by other jobs or they may not, or the jobs replacing them may be more poorer paid than better paid in this economic phase, the opposite of the situation in the last one. If the size and prosperity of this information handling middle class has been reduced so much, and this middle class and its money has fueled a demand for a certain quality and quantity of consumer good, then the consumer goods that the retail, etc. sector has been furnishing may no longer find a market, or certainly not the same or as big a market. Similarly, those oriental companies and governments that have invested in manufacturing plants on the assumption that the capital debt could be paid off by the same level of demand and high priced product that the purchases of this once larger and more prosperous American middle class used to make, may have a lot of bad debts on their hands in the future. A low wage manufacturing country used to gradually become a higher wage country, but this pattern may not be repeated tomorrow. It is obviously a very big change in economic structure we are seeing here, not a blip.

At the grass roots level, where you and I live, what can this mean? For one thing if the deindustrialization has not yet really hit (or may never), this alone will put a different answer to the question of where the future jobs will be found. Personal service is likely to be the buzz word for the lower level jobs, restaurant, assisting helping make choices and customize purchases and services, right down to social, etc. secretaries and housekeeping. On the higher level, it will be the age of the consultant if it is not already. This certainly means that technical and professional people will need a broader range of skills to be in this line of work, for like a farmer, a consultant will have to keep the books, fix the office chair and computer connections, type the bills and calculate the amounts, sell the services in the first place and do the promotion and public relations and complaint adjustment. We are not talking of the generalist instead of the specialist, but the generalist skills needed to be added to those of the specialist. One cannot know too many different things today.

Finally, jobs will be found that do not exit today, just as word processor and CAD and desktop publishing jobs did not exist in any number, just a dozen years ago.

---

* Daniel Bell on Post-Industrial Societies

Disks, Cassettes, Modems, Older Home Computers - Some New Tips, Problems

Floppy disks used to be fairly expensive and both the disks and the disk drives employed with older computers in their heyday often were of poor quality making for many unexpected problems for the users. A lot of those older, clunkier disk drives are still around, many original ones and many bought by hardware tinkerers to add to their computers. The older drives may not like to write to the innermost tracks for example, or spoil disks easily during normal operation, or be out of alignment and make saves to disks that no other computer with a normally aligned disk drive can read, so you cannot swap disks with them. Similar alignment incompatibilities may plague cassette use. Some the older computers that use cassettes have many peculiarities in what they need (a fresh cassette rather than a merely erased cassette to store programs on, frequent cleaning, etc.). So use of one of these older computers may be a problem if you are not a born tinkerer. Otherwise the more expensive way to fix it by simply replacing the disk drive or cassette unit by a brand new one can be done You may have to ask around a user group to find someone to do it or a source for buying a suitable unit in the case of some makes and models nonetheless.

These problems will be something that old users are familiar with so that a great deal of space will not be devoted to discussing these points here. They often have been covered in old, user group newsletters available through user group libraries, and old magazines for the make of computer that you have.

However some new problems are arising now. Some disks that are being sold now work perfectly all right with newer computers but not with older ones. It seems that the operating systems of newer computers often are better able to cope with small imperfections in the disks. For example, if a disk has a fault on the track that the directory is located on, this may not bother a computer that uses an operating system that puts two directories on the disk or file tables (FATS) so even if one is impaired by a bad magnetic fault, the other which is a duplicate copy, will be all right. If the operating system on your computer only lokks at one of the copies of the directory track info, or only writes info to one of them anyway, then you may have a problem with a cheaper floppy disk that a newer computer may not notice at all. This would seem to be the case with some disks used even in IBM compatible computers running MS DOS versions earlier than 2 compared to those running versions numbered 2 or higher. Any other computer of an older vintage that uses track 0 for its directory info may have the same problem since older operating systems rarely made duplicate copies of the directory information on each floppy disk as a matter of course. But those computers or operating systems used with some computers that put the directory tables on other tracks (track 18 for Commodore 64, track 17 for Radio Shack Color Computer BASIC O/S disks) may encounter this problem if the flaw turns up on their directory track. So if you take the disks back to the store and the sales or service personnel cannot find any problem when they use the disks in their computers that does not prove anymore that there is something wrong with your computer or nothing is wrong with the disks. For my own use of older computers, stopping buying generic disks or cheaper ones, and sticking with brands of known quality like BASF has become my current practice as a result. The bad disks will probably run on an IBM PC clone with MS DOS version 2 or higher so you can save them for that use. It is computer and OS variable often.

Something similar seems to affect modem use with older modems that run at slower speeds. Some areas have telephone systems that are poor enough that a lot of errors get into transmissions. Newer software can often handle this and since newer modems run at much faster speeds, the software has the time to let the modem program writers add extra routines to retransmit blocks, etc. Older software used with older computers may not have this. But running a faster modem, with software that cannot cope with the errors of a noisy line may just make the problem worse, since high speeds are harder to run error free on bad lines, other things being equal. Using a better quality modem may help if one comes your way to experiment with and some hardware hackers have tried filters with mixed success (and worry about zapping something). Some amateurs may also have forgotten that accoustic coupled modems are not good for long transmissions with phones that use the old carbon microphones, although newer units that use electret mics inside may fix that problem (and telephones are so cheap now that trying a new one is no great expense). This all means user groups are still needed to swap tips and fixes on these older, classic, home computers. 1508-92

Bug Reports And Other ZX-TS Ponderings

When you are a beginner or just a creative hobbyist trying to delve deeper into how computers work, you can easily get fooled by believing that since an expert produced the book, it cannot be wrong if something does not make sense. Well experts cannot be experts in everything and they can make mistakes. Witness a magazine article, some technical books by professors and a book on OS/2 which got some basic facts wrong to count about a dozen errors this writer has found recently in print. For the Sinclair enthusiast, here are a few more of interest

old ZX-81 manual, p.184 (not in newer TS manual) code 135 is ▗, not ▛

in the ZX-81 LDOS disk manual sort is 14480 and CLINE is 12316 or should.

old ZX-81, 2 volume version of Dr.Logan's ROM disassembly there are a number of misprints in addresses, so before calling one in a m/c call routine, look at the previous and next address to see if it makes sense (as in p.2, both FP-CALC and the END-CALC following are given the same ROM address 0028 hex, while the second should be 002B). All m/c addresses of any book are suspect!

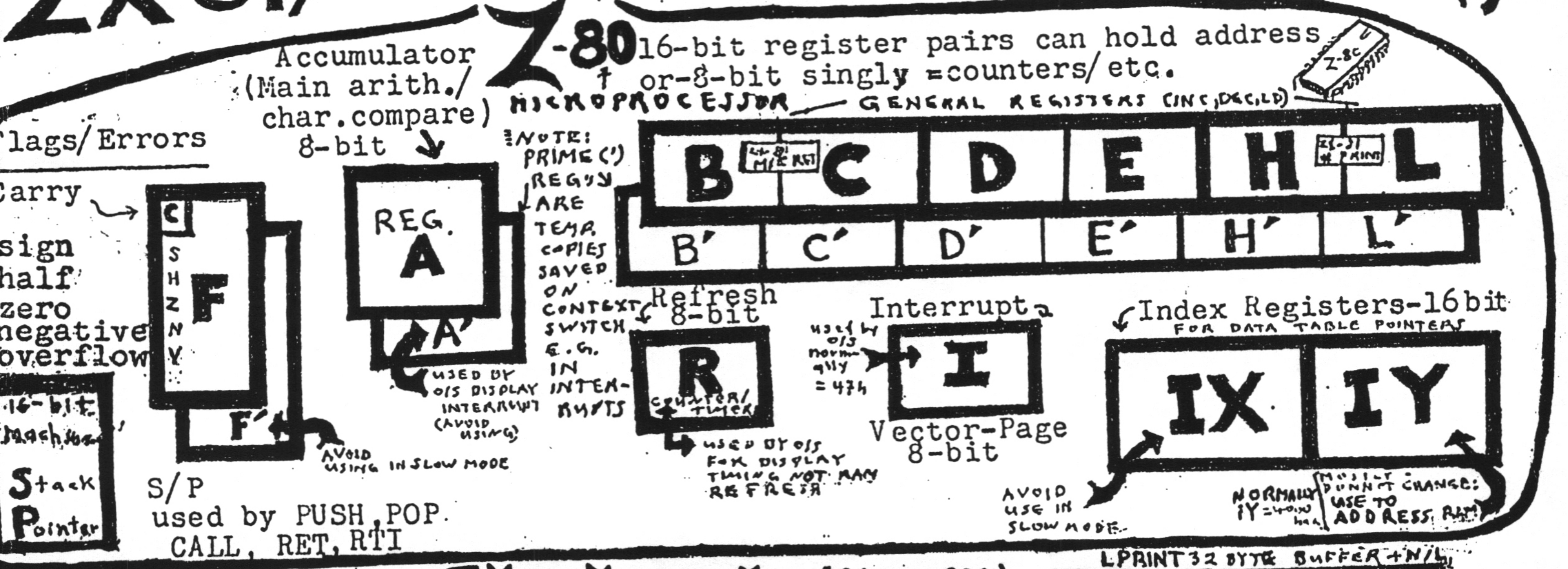
Not to be outdone, our LDOS Unofficial Technical Manual has included a few (this is a ZX-81/TS1000 only book), in the different versions that have come out. Even some very early versions of the simple instructions on using the system for beginners, contained some mistaken advice, that was pointed out to me by early reviewers. Also, picking up reports that subway train motors can erase disks was a mistake that was pointed out, the story possibly originating as an imaginitive excuse to someone's boss to explain why the work to be done at home was not on the boss' desk the next morning. One of those urban myths perhaps. A specific error in the section on converting commercial programs (page 3-4 in some versions) quoted SAVE CHR$... instead of LOAD CHR$ 832"" as the way to get an unbreakable program to break, the break key halting it when it pauses after loading before going into a save. Another mistake was in saying that you could PEEK to CLINE after an operation to get the file name last saved under, when in fact this is partially overwritten with temporary data during the course of disk access. This publishing hobby can certainly keep one sharp.

Then there are problems in the ZX-81/TS1000 itself. The occasional error with floating point divide is fairly well known, but some early ROM versions will make other mistakes like giving 1.359 for PRINT SQR 0.25, when everyone knows the square root of 1/4 should be something like 0.5 ($\frac{1}{2}$ times $\frac{1}{2}=\frac{1}{4}$). It seems that Sir Clive Sinclair was an enthusiast for finding new, smaller m/c programs for getting math functions, having been able to squeeze some scientific functions into the small ROM of a pocket calculator, in a way people thought could not be done, and doing it in record time from a motel room in the USA near the chip manufacturer's plant, that was waiting to start production. The most obvious mistake in the ZX-81/TS1000 ROM is the CLS going nuts when used after some scrolling, in certain situations. However the Sinclairs were not the only computer to have some software bugs, as MS DOS in versions below 2 and the BASICA that goes with it had enough bugs that owners were advised to upgrade to version 2 in the early days of the IBM PC. One can probably consider oneself lucky if one only makes a few errors in a project as exacting as computer work. The Coleco Adam computer for example has had its BASIC fixed extensively by local amateurs working out of the user groups. One problem it once had, is forgetting which parts saved on a cartridge or disk belong to the latest or an older, deleted version of an essay or text, to lose the whole thing as a result, not a nice thing when you are working to a deadline. (And would the teacher believe you if you said it was all finished, but the computer lost it? Maybe he would take the subway motor erased it story as more believable afterall.)

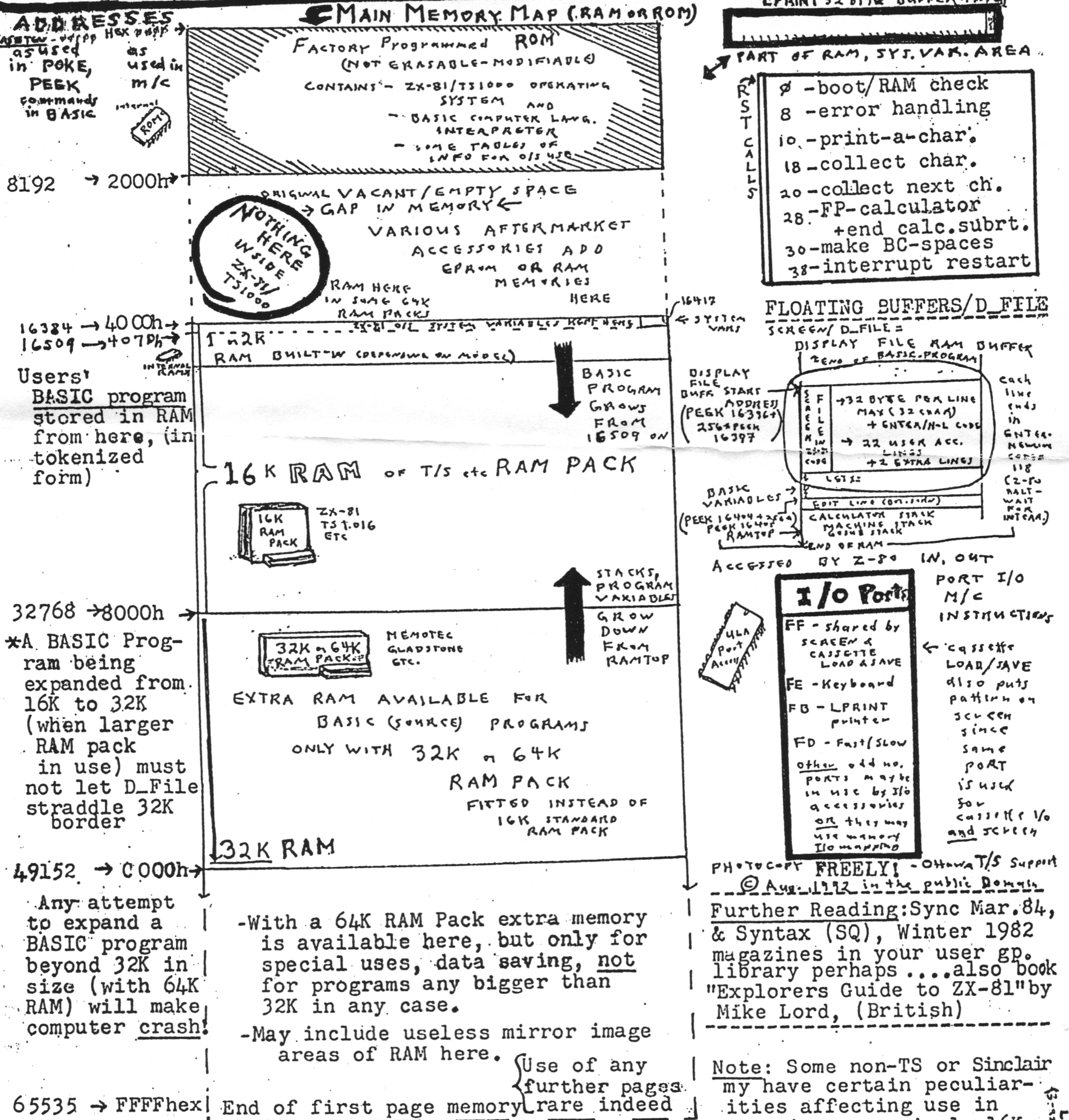
Finally, in the last issue (TS Bulletin #2-92) an indexed sorting routine was listed for educational purposes. It occurred to me that RAM could easily be saved by replacing the array N(X), DIM N(NE) of numbers with the string array N$(X), so that DIM N$(NE,1) would save 4 bytes per element (record, in the database). This would limit records to about 253 and the comparisons etc. would have to be changed to IF OP AND I$(CODE(N$(X)))(3 TO 23)<=I$(CODE((N$(Y))))(3 TO 23) THEN GOTO 225 from 202 IF OP=1 AND I$(N(X))(3 TO 23)<=I$(N(Y))(3 TO 23) THEN GOTO 225, and similar changes to the other comparisons, etc. and switch routines at 210 and 280. 110 also becomes 110 LET N$(Z)=CHR$ Z etc.

So look out for our little friend the computer bug; he is everywhere!

PROGRAMMERS MAP OF ZX-81/TS1000 HOME COMPUTER


Z-80†
MICROPROCESSOR
Accumulator (Main arith./char.compare) 8-bit
REG. A
A'
USED BY O/S DISPLAY INTERRUPT (AVOID USING)
Flags/Errors
Carry
sign
half
zero
negative
overflow
C S H Z N V
F
F'
AVOID USING IN SLOW MODE
16-bit Mach.
Stack Pointer
S/P used by PUSH, POP, CALL, RET, RTI
NOTE: PRIME (') REGS ARE TEMP. COPIES SAVED ON CONTEXT SWITCH E.G. IN INTERRUPTS
GENERAL REGISTERS (INC,DEC,LD)
B C D E H L
B' C' D' E' H' L'
M/C RET
PRINT
Z-80
Refresh 8-bit
R
COUNTER/TIMER
USED BY O/S FOR DISPLAY TIMING NOT RAM REFRESH
Interrupt
used by O/S normally = 47h
I
Vector-Page 8-bit
Index Registers-16bit
FOR DATA TABLE POINTERS
IX IY
AVOID USE IN SLOW MODE
NORMALLY IY=4000 hex
MUST NOT CHANGE: USE TO ADDRESS RAM

16-bit register pairs can hold address or-8-bit singly =counters/etc.


ADDRESSES
DECIMAL as used in POKE, PEEK commands in BASIC
HEX as used in m/c
internal ROM
MAIN MEMORY MAP (RAM OR ROM)
FACTORY Programmed ROM
(NOT ERASABLE-MODIFIABLE)
CONTAINS - ZX-81/TS1000 OPERATING SYSTEM AND
- BASIC COMPUTER LANG. INTERPRETER
- SOME TABLES OF INFO FOR O/S USE
8192 → 2000h
ORIGINAL VACANT/EMPTY SPACE
→ GAP IN MEMORY ←
NOTHING HERE INSIDE ZX-81/TS1000
VARIOUS AFTERMARKET ACCESSORIES ADD EPROM OR RAM MEMORIES HERE
RAM HERE IN SOME 64K RAM PACKS
16384 → 4000h
16509 → 407Dh
INTERNAL RAM
ZX-81 O/S SYSTEM VARIABLES KEPT HERE
16417
SYSTEM VARS
1 or 2K RAM BUILT-IN (DEPENDING ON MODEL)
BASIC PROGRAM GROWS FROM 16509 ON
16K RAM OF T/S etc RAM PACK
16K RAM PACK
ZX-81 TS 1016 ETC
STACKS, PROGRAM VARIABLES GROW DOWN FROM RAMTOP
32768 → 8000h
32K or 64K RAM PACK
MEMOTEC GLADSTONE ETC.
EXTRA RAM AVAILABLE FOR BASIC (SOURCE) PROGRAMS ONLY WITH 32K or 64K RAM PACK FITTED INSTEAD OF 16K STANDARD RAM PACK
32K RAM
49152 → C000h
65535 → FFFFhex
LPRINT 32 BYTE BUFFER + N/L
PART OF RAM, SYS. VAR. AREA
RST CALLS
FLOATING BUFFERS/D_FILE
SCREEN/D_FILE = DISPLAY FILE RAM BUFFER
END OF BASIC PROGRAM
DISPLAY FILE BUFF START ADDRESS (PEEK 16396+256*PEEK 16397)
SCREEN FILE IN ZX81 CODE
→ 32 BYTE PER LINE MAX (32 CHAR) + ENTER/N-L CODE
→ 22 USER ACC. LINES + 2 EXTRA LINES
each line ends in ENTER/NEWLINE CODES 118 (Z-80 HALT-WAIT FOR INTERR.)
LIST:
BASIC VARIABLES
EDIT LINE (ON SCREEN)
CALCULATOR STACK
MACHINE STACK
GOSUB STACK
(PEEK 16404+256*PEEK 16405) RAMTOP
END OF RAM
ACCESSED BY Z-80 IN, OUT PORT I/O M/C INSTRUCTIONS
ULA Port Access
I/O Ports
FF - shared by SCREEN & CASSETTE LOAD & SAVE
FE - Keyboard
FB - LPRINT printer
FD - Fast/Slow
other odd no. PORTS may be in use by I/O accessories OR they may use memory I/O mapping
← cassette LOAD/SAVE also puts pattern on screen since same PORT is used for cassette I/O and screen

Users' BASIC program stored in RAM from here, (in tokenized form)

*A BASIC Program being expanded from 16K to 32K (when larger RAM pack in use) must not let D_File straddle 32K border

Any attempt to expand a BASIC program beyond 32K in size (with 64K RAM) will make computer crash!

-With a 64K RAM Pack extra memory is available here, but only for special uses, data saving, not for programs any bigger than 32K in any case.

-May include useless mirror image areas of RAM here.

End of first page memory

Use of any further pages rare indeed

RST CALLS:

0 -boot/RAM check
8 -error handling
10 -print-a-char.
18 -collect char.
20 -collect next ch.
28 -FP-calculator +end calc.subrt.
30 -make BC-spaces
38 -interrupt restart

PHOTOCOPY FREELY! - Ottawa T/S Support

© Aug. 1992 in the public Domain

Further Reading: Sync Mar. 84, & Syntax (SQ), Winter 1982 magazines in your user gp. library perhaps ....also book "Explorers Guide to ZX-81" by Mike Lord, (British)

Note: Some non-TS or Sinclair my have certain peculiarities affecting use in certain cases, incl. 16K.

---

†Z-80 is a registered name of Zilog Inc.

* Care Needed: When you are writing a BASIC program, when it is just filling the first 16K and getting to the point where it is starting to be bigger than a 16K RAM pack equipped computer can hold, you must avoid the gradual expansion that puts the D_File accross this boundary. (Duplicating a giant REM is usual fix.)